Title: Trinity Burst Buffer - Hardware and Software Architecture

Author(s): Wright, Cornell G. Jr.

Intended for: Trinity Open Science, 2015-10-07 (Los Alamos, New Mexico, United States)

Issued: 2015-10-07

**Disclaimer:**
Los Alamos National Laboratory, an affirmative action/equal opportunity employer,is operated by the Los Alamos National Security, LLC for the National NuclearSecurity Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Departmentof Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

# Trinity Burst Buffer

## Hardware and Software Architecture

Cornell Wright

Nathan Hjelm

October 2015

UNCLASSIFIED

# Agenda

- Acknowledgements
- Burst Buffer Background
- Trinity Burst Buffer Hardware and Software
- HIO Design
- DataWarp Administration
- Multi-Job Scenarios
- Project Status

# Acknowledgements

- Many people and groups have contributed to the concept, design and development of burst buffer, Trinity and the materials in this talk.
- Including:
  - Gary Grider
  - Josip Loncaric
  - Doug Doerfler (SNL)
  - Nathan Hjelm
  - Nick Wright (NERSC)
  - Dave Henseler (Cray)
  - Bob Pearson (Cray)
  - Bronis de Supinski (LLNL)
  - Adam Moody (LLNL)
  - John Bent (EMC)

# Agenda

- Acknowledgements
- Burst Buffer Background
- Trinity Burst Buffer Hardware and Software
- HIO Design
- DataWarp Administration
- Multi-Job Scenarios
- Project Status

# Terminology

- Burst Buffer
  - A high-speed, low cost-per-bandwidth storage facility used to reduce the time spent on high volume IO.
- DataWarp™
  - A Cray SSD storage product that provides burst buffer (and other) function. Initially developed for Trinity and Cori.
- Hierarchical IO Library (HIO)
  - A LANL developed API and library which facilitates the use of burst buffer and PFS for checkpoint and analysis IO on Trinity and future systems.

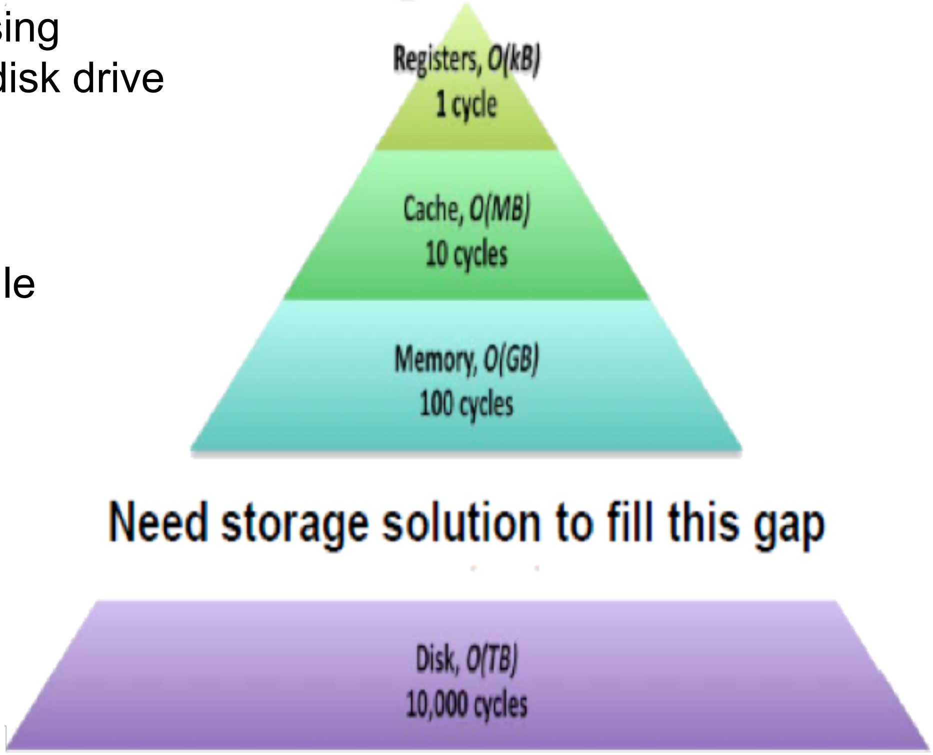
# Burst Buffers will improve Productivity and Enable Memory Hierarchy Research

- Technology Drivers:
  - Solid State Disk (SSD) cost decreasing
  - Lower cost of bandwidth than hard disk drive

- Trinity Operational Plans:
  - SSD based 3 PB Burst Buffer
  - 3.28 TB/Sec (2x speed of Parallel File System)

- Burst Buffer will improve operational efficiency by reducing defensive IO time

- Burst Buffer fills a gap in the Memory and Storage Hierarchy and enables research into related programming models



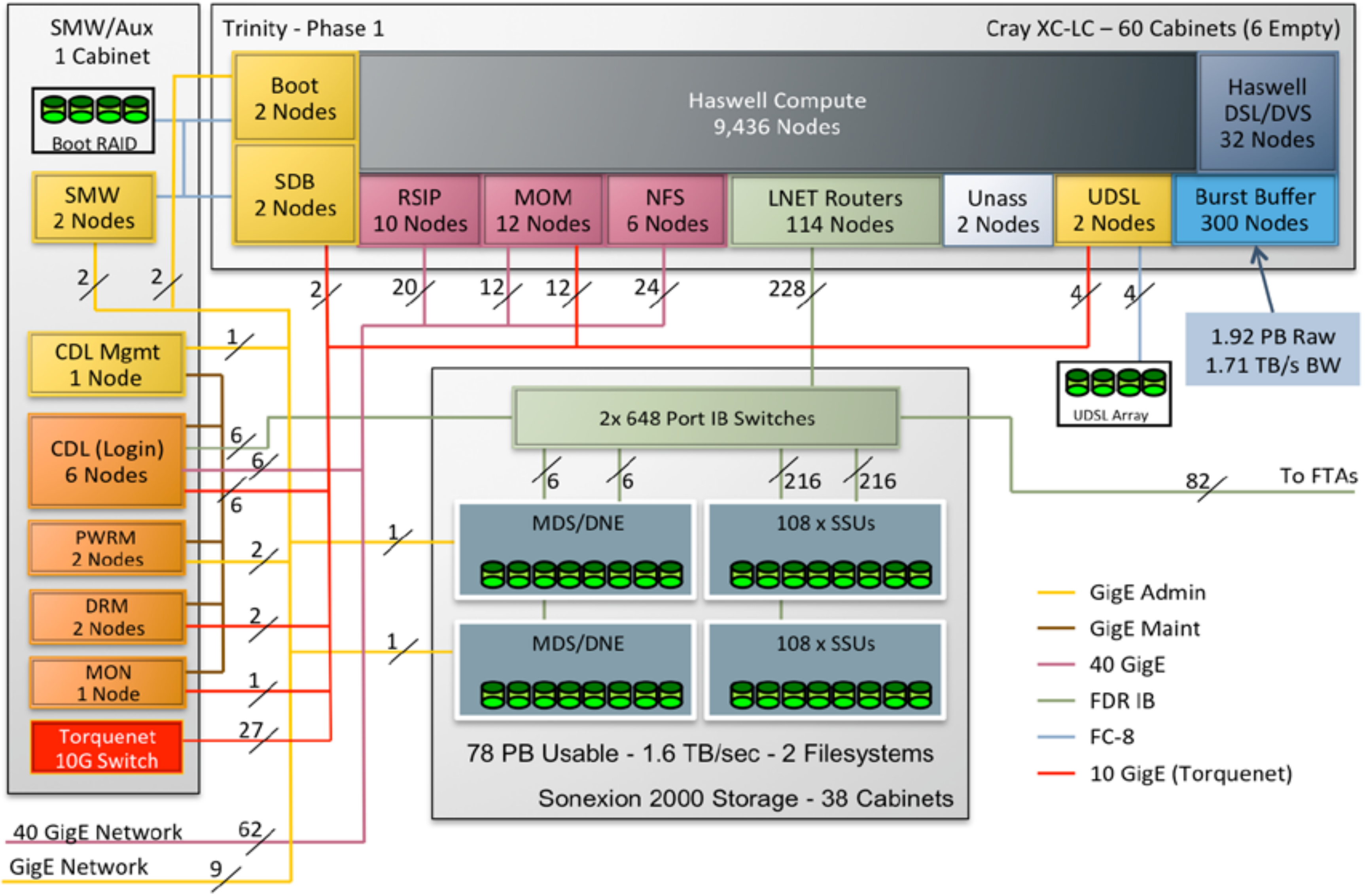
# Burst Buffer – more than checkpoint

- Use Cases:
  - Checkpoint
    - In-job drain, pre-job stage, post-job drain
  - Data analysis and visualization
    - In-transit
    - Post-processing
    - Ensembles of data
  - Data Cache
    - Demand load
    - Data staged
  - Out of core data
    - Data intensive workloads that exceed memory capacity

# Agenda

- Acknowledgements
- Burst Buffer Background
- Trinity Burst Buffer Hardware and Software
- HIO Design
- DataWarp Administration
- Multi-Job Scenarios
- Project Status

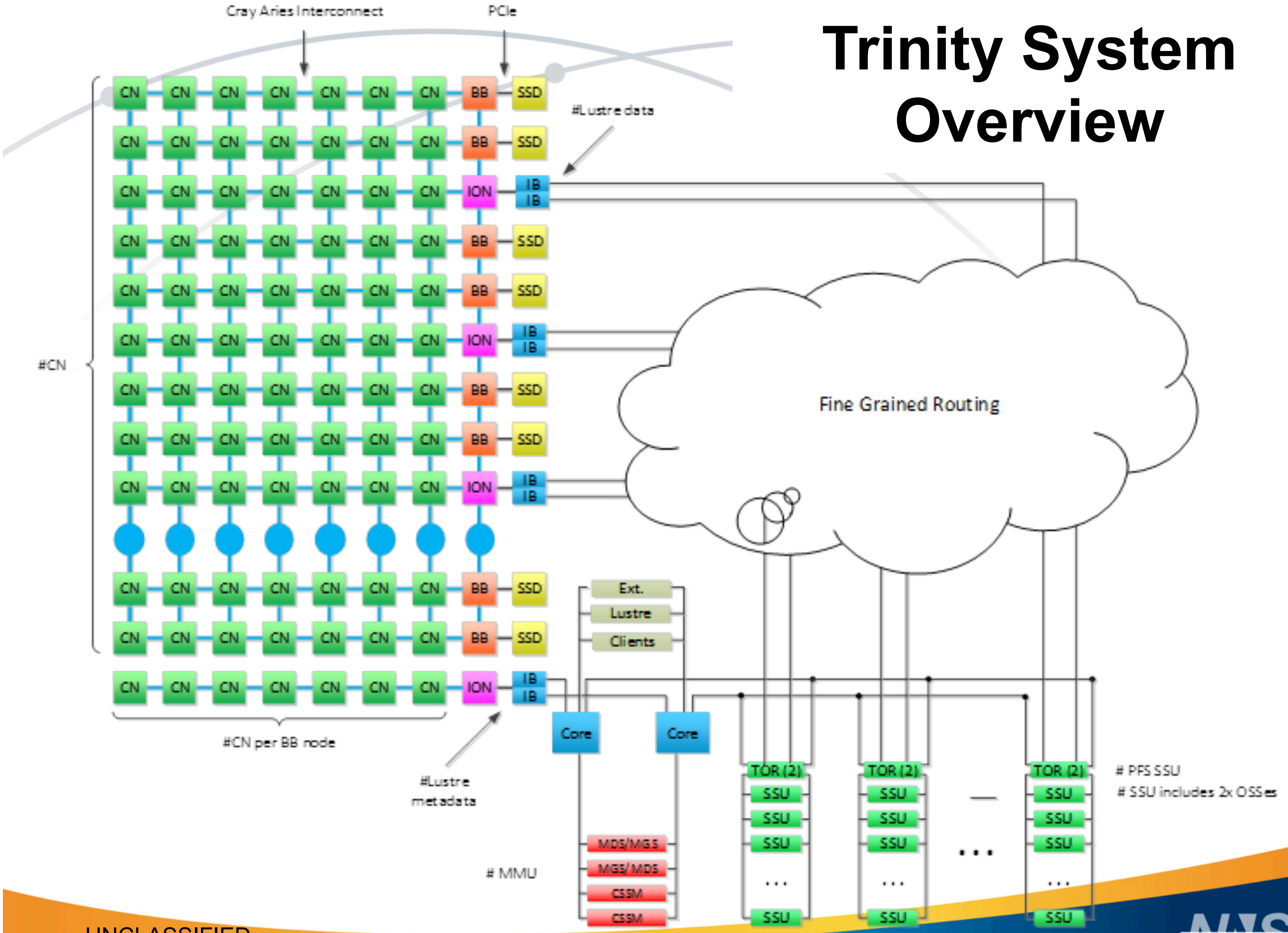
# Trinity Phase 1 Configuration



# Trinity by the Numbers

| Parameter | Phase 1 | Phase 2 | Total |
|---|---|---|---|
| Nodes | ~9500 Haswell | ~9500 KNL | ~19000 Nodes |
| Cores/Node | 32 | 64 - 72 | |
| HW Threads/Node | 64 | 256 - 288 | |
| Memory/Node | 128 GiB | 96 GiB (+16G HBM) | |
| Total Memory | ~1.15 PiB | ~0.91 PiB | ~2.07 PiB |
| Node Peak Perf | ~1.18 Tflops | ~3.01 Tflops | |
| System Peak | ~11.1 Pflops | ~30.7 Pflops | ~41.8 Pflops |
| PFS Capacity | 78 PB | Unchanged | 78 PB |
| PFS Bandwidth | ~ 0.8 TB/S | ~ 0.8 TB/S | 1.6 TB/S |
| Burst Buffer Nodes | 300 | 276 | 576 |
| BB Capacity | 1.92 PB | 1.77 PB | 3.65 PB |
| BB Bandwidth | 1.71 TB/S | 1.57 TB/S | 3.28 TB/S |

# Trinity System Overview



UNCLASSIFIED

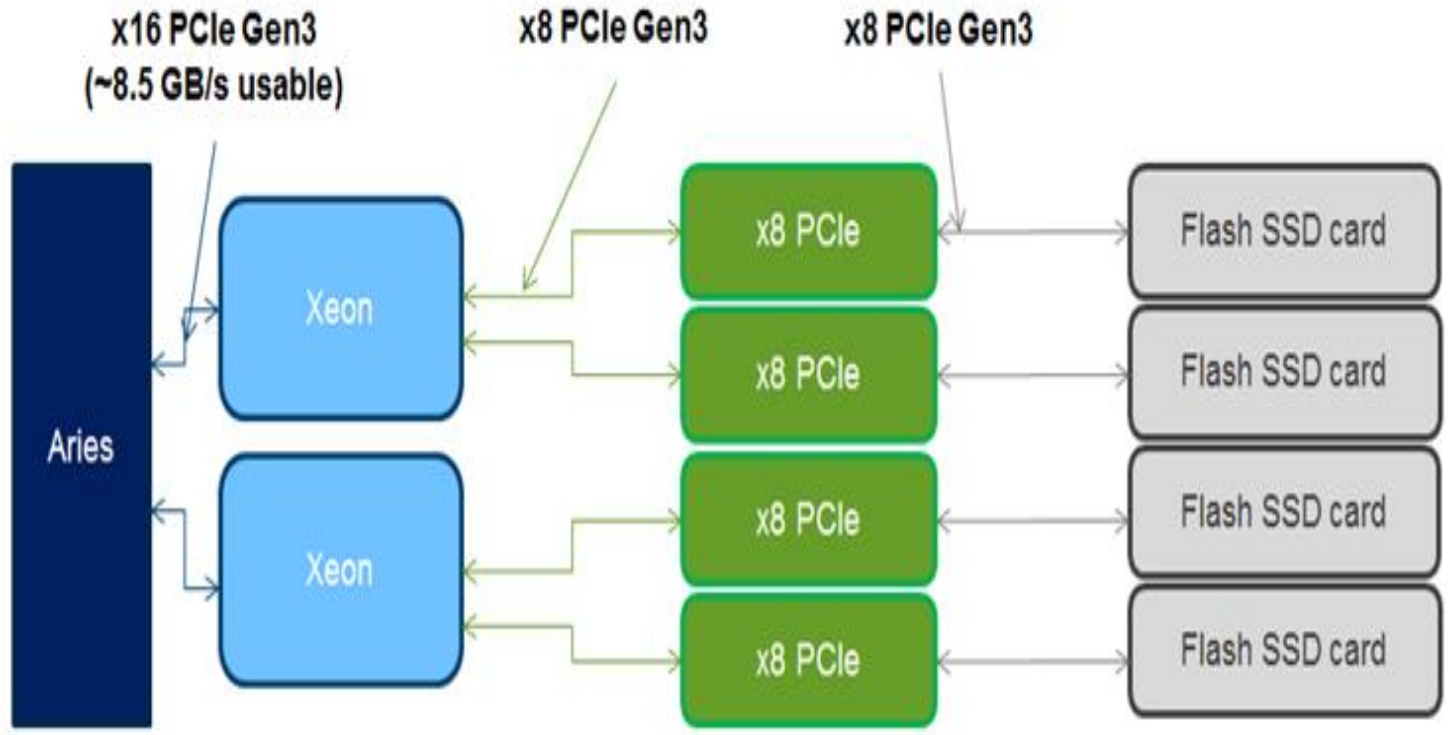
# Trinity Burst Buffer Hardware

- 576 Burst Buffer Nodes
  - Announced as Cray DataWarp™
  - On high speed interconnect - globally accessible
  - Trinity IO Node + PCIe SSD Cards
  - Distributed throughout cabinets

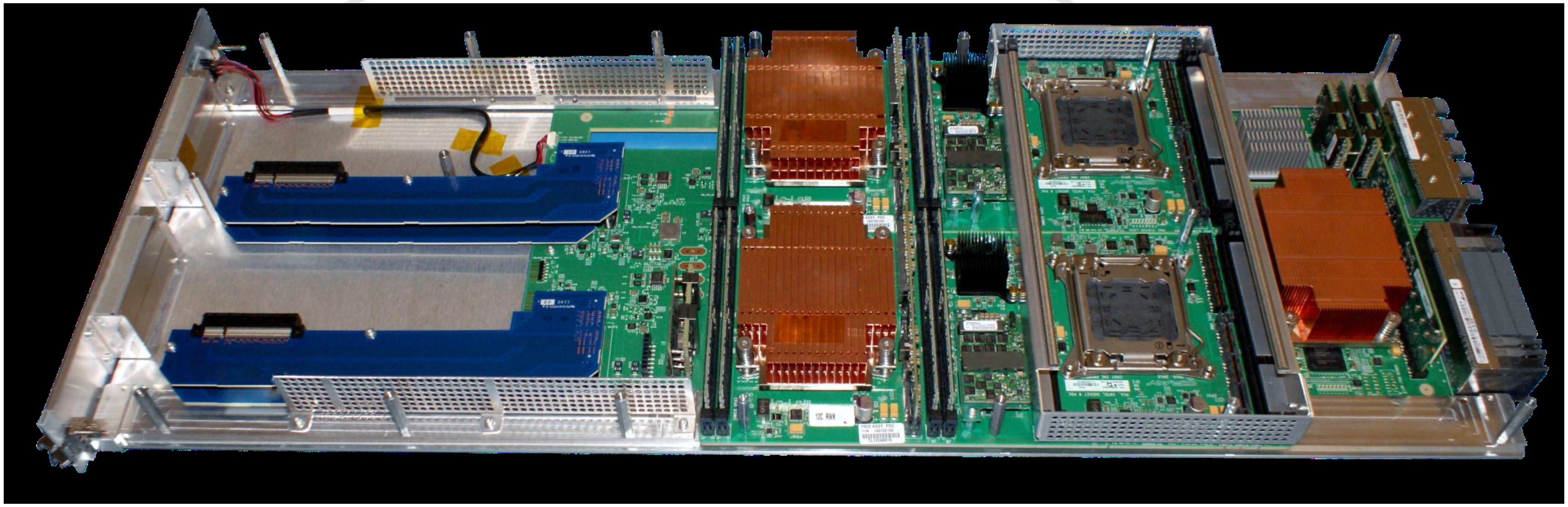
| Metric | Phase 1 | | Full Trinity | |
|---|---|---|---|---|
| | Burst Buffer | PFS | Burst Buffer | PFS |
| Nodes | 300 BB Nodes | 114 LNET Routers | 576 BB Nodes | 234 LNET Routers |
| Bandwidth | 1.7 TB/S | 0.71 TB/S | 3.3 TB/S | 1.45 TB/S |
| Capacity | 1.9 PB | 82 PB | 3.7 PB | 82 PB |
| Memory Multiple | 1.6 X | 67 X | 1.75 X | 39 X |
| Full system checkpoint cost | | | 12% | 21% |

# Cray XC40 DataWarp™ Blade

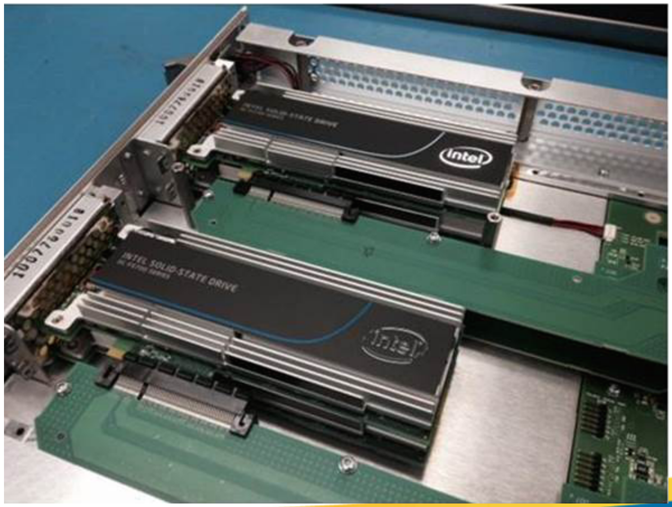
( 2 Burst buffer Nodes )



# Cray DataWarp™ Blade



↑ Service Blade
(2 nodes)



← SSD Cards

# DataWarp Modes

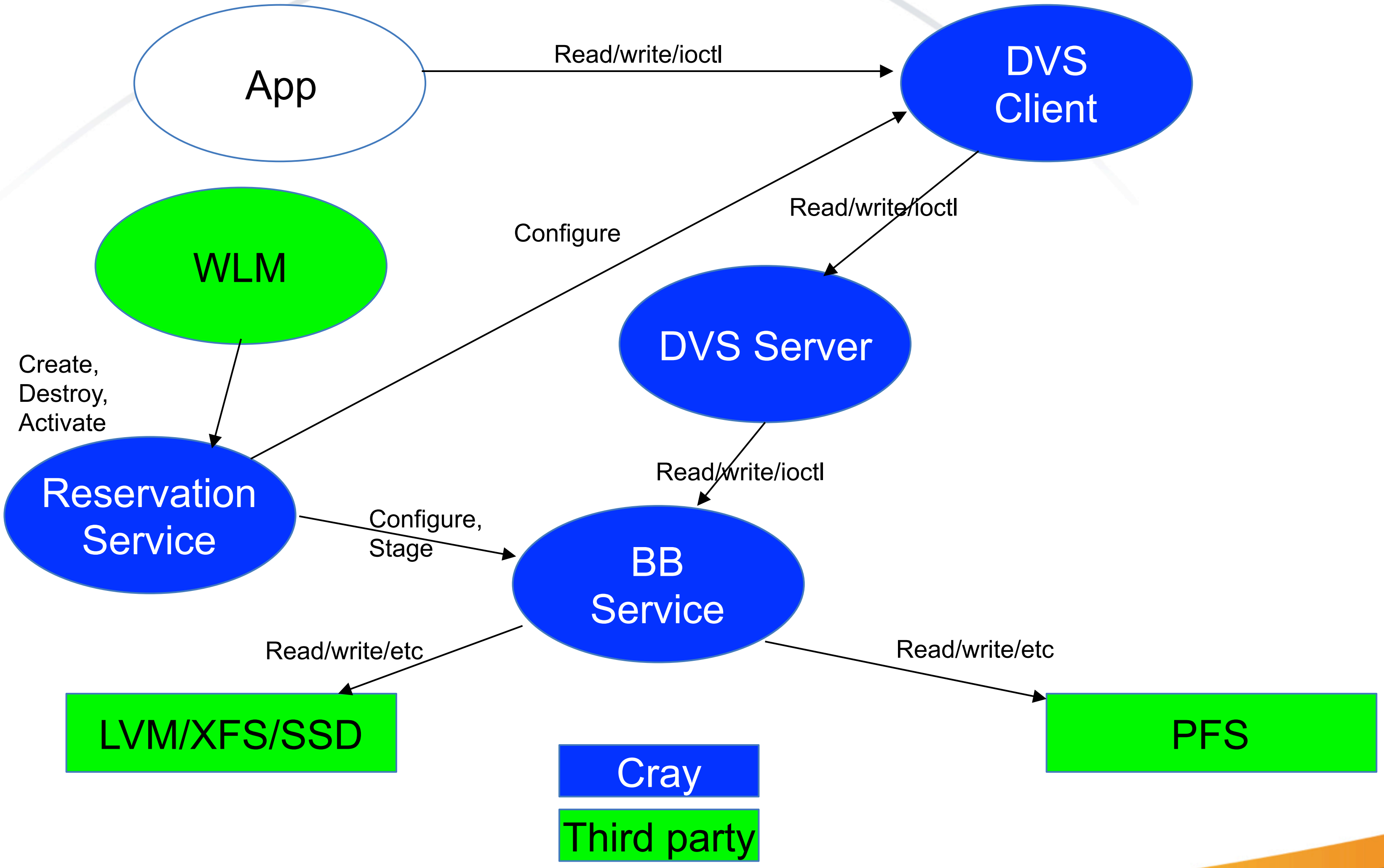
| Mode | Description |
|---|---|
| Private Scratch | Per node burst buffer (BB) space |
| Shared Scratch | Shared space, files may be striped across all BB nodes. → Used for Trinity Checkpoints ← |
| Shared Cache | Parallel File System (PFS) cache.<br>Transparent and explicit options |
| Load Balanced Read Only Cache | PFS files replicated into multiple BB nodes to speed up widely read files |

UNCLASSIFIED

# DataWarp System Software

- DataWarp SSD partitioned into allocation units
  - Allocation units belong to LVM volume group
- Workload manager
  - Job submission requests DW capacity
  - Starts job when capacity available
- DataWarp registration service
  - Selects allocation units
  - Creates XFS logical volumes on SSD
  - Mounts via DVS on compute nodes
- Automated stage/drain of specified directories from/to PFS
- Per job write limits (endurance management)
- Administrative Functions – configuration, monitoring, repair

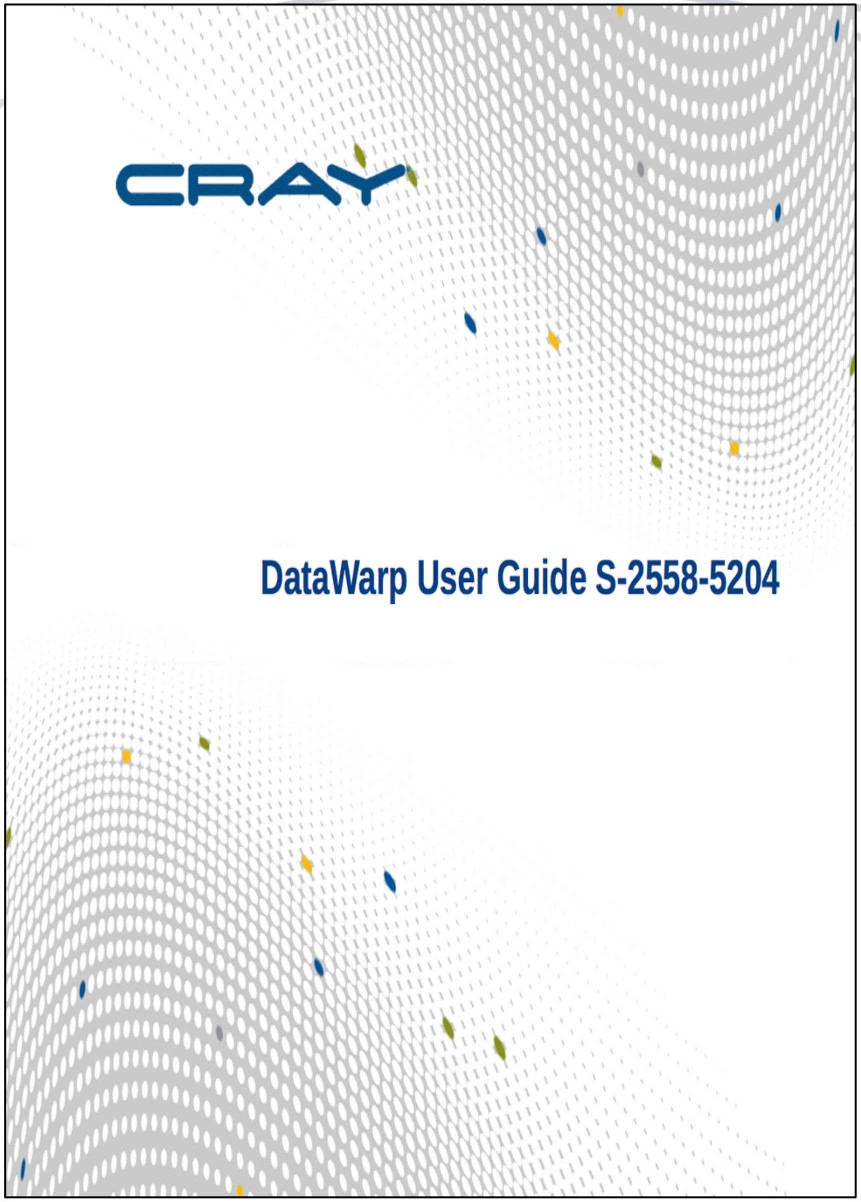
# Burst Buffer System Software



# Workload Manager Integration

## DataWarp Directives in Job Script

- #DW jobdw access_mode=mode capacity=n type=scratch
  - Creates a job DataWarp instance for this job
- #DW persistentdw name=piname
  - Configures access to an existing persistent DataWarp instance for this job
- #DW stage_in destination=dpath source=spath type=type
  - Stages files from the PFS to the DataWarp instance before the user job starts
- #DW stage_out destination=dpath source=spath type=type
  - Stages files from the DataWarp instance to the PFS after the user job completes
- #DW swap nGB
  - Configures compute node swap for this job
  - Not yet available (ETA Nov 15)


CRAY

DataWarp User Guide S-2558-5204

- DataWarp User Guide
  - Overview and Concepts
  - Administrative Commands
  - Job Script Commands
  - DataWarp API
  - 48 Pages

# Agenda

- Acknowledgements
- Burst Buffer Background
- Trinity Burst Buffer Hardware and Software
- HIO Design
- DataWarp Administration
- Multi-Job Scenarios
- Project Status

# Applications have Options

- Hierarchical Input Output (HIO) Library
  - Hides vendor specific interface
  - Provides additional performance, reliability, management and diagnostic functions
  - Recommended approach
- Scalable Checkpoint Restart Library (SCR)
  - Adam Moody (LLNL) planning to support Trinity BB with SCR
- Direct POSIX calls from application
  - Will require Cray DataWarp specific ioctl calls to exploit striping and stage / drain functions

# HIO Design Goals

HIO, short for Hierarchical Input Output, is the burst buffer enablement library for Trinity and other systems.

Goals:

- Support Trinity and future Burst Buffer implementations
  - Isolate application from BB technology shifts
- Easy to incorporate into existing applications
  - Lightweight, simple to configure
- Improve checkpoint performance
- Improve job reliability and efficiency
- Support checkpoint and analysis scenarios (e.g., viz)
- Extend to bridge to future IO
  - e.g., Sierra, HDF5, IO Fast Forward II

# Why implement HIO as a library ?

- Simplest packaging and delivery available
- Self contained, minimal external or system dependencies
- Easiest for applications to adopt
- Library approach facilitates rapid prototyping and deployment, responsiveness to application needs
- Library also provides a vehicle to provide (at no cost to applications):
  - Checkpointing best practices
  - Performance and functional diagnostics
  - Mitigation for system problems
- Why not provide via extensions to MPI-IO now ?
  - Existing implementation perform poorly
  - Unloved by users
- Integration with other IO packages will be investigated in future
  - HDF5 ? SCR ? ADIOS ?
  - HIO library should be largely reusable in that environment

# HIO Project Features

- Full thread safety
- C/C++/Fortran (2008) support
- Configurable diagnostic and performance monitoring
- Header / Library packaging
- Open-source intention
- Support tri-lab ATS and CTS systems (more than Trinity)
- Prototyped EAP support for HIO as POC and test vehicle
- PFS-only version available now

# HIO Design Features

- Flexible configuration capability
- Abstract view of IO namespace
- Open/Read/Write/Close data interfaces
- Checkpoint management
- Hardware error recovery
- (Future) Job management

# HIO – Flexible Configuration

- Keyword=value format
- Multiple sources for flexibility
  - System file (optional)
  - Application file (entire or partial)
  - Unix environment
  - API call
- Applied on rank 0 and propagated

# HIO – Abstract Namespace

- Named Context / Dataset / ID / Element
  - Context: All data managed by an HIO Instance
  - Dataset: Particular type/format of data
  - ID: Instance of data, expected to be sequence
  - Element: Named section of dataset
- On Trinity, will map to BB & PFS directory structure
- Future system’s BB may not have FS
- Basic mode writes directly to N-N or N-1 file(s)
- Optimized mode:
  - Data restructured for performance
  - Transparent to application
  - Limited guarantees on physical file structure
  - Will provide transfer utility to/from N-N or N-1 files

# HIO – Data Interfaces

- Open Dataset specific ID or highest ID
- Synchronous or asynchronous read and write
- Optional strided read and write
- Performance features:
  - Turnstile to limit concurrency
  - Multiple destination directories
- Possible future directions:
  - HDF5 interface
  - POSIX read / write intercept
  - MPI-IO implementation
  - ADIOS interface

UNCLASSIFIED

# HIO – Checkpoint Management

- Interface recommends checkpoint based on:
  - System Characteristics:
    - System and Node MTTI
    - BB Bandwidth
  - Job Characteristics:
    - Job size
    - Checkpoint size
    - Application Reliability
- Periodic BB checkpoint background drain to PFS
- BB checkpoint deletion for space management

# HIO – Hardware Error Recovery

- Multiple data roots
  - e.g., BB; /scratch1; /scratch2
  - Errors on read handled by notification and subsequent fallback to secondary (or tertiary) root
  - Errors on write handled by notification, potential fallback to secondary (or tertiary) root and immediate rescheduling of checkpoint
  - Active data root's bandwidth will influence recommended checkpoint interval
- Any complete BB checkpoint will be marked eligible for post job drain to PFS (even if job subsequently fails.)
- Application level CRC on data (optional)

# HIO – Job Management

- Potential future capabilities:
  - Enable graceful shutdown of jobs and system with final checkpoint to PFS
  - Schedule PFS traffic to reduce contention

# HIO - Instrumentation

- Delivered via:
  - API calls (hio_perf_xxxxxx)
  - Messages to stdout (controlled via configuration)
- Values:
  - Read/write byte counts
  - Operation counts
  - Average size
  - Time elapsed
  - Speed
  - Etc.
- Extend as user and application analysis needs require

# HIO Basic and Optimized Mode

- Basic Mode:
  - Application reads and writes performed directly on underlying file system
  - File structure preserved
  - Available now
- Optimized Mode
  - Application reads and writes shipped to designated IO nodes
  - Builds on xRage bulkIO concepts
  - File count, layout and striping optimized
  - Exploits modern MPI one-sided message capability
  - N-M IO pattern (when needed)
  - Code complete end September
- Same application interface, selected via configuration

# Primary HIO API Calls

- Init / Term / CP Mgmt:
  - hio_init_mpi()
  - hio_config_set()
  - hio_should_checkpoint()
  - hio_fini()
- Open / Close:
  - hio_dataset_open()
  - hio_element_open()
  - hio_element_close()
  - hio_dataset_close()
- Read / Write:
  - hio_element_write {strided}{nb}()
  - hio_element_read {strided}{nb}()
  - hio_wait()

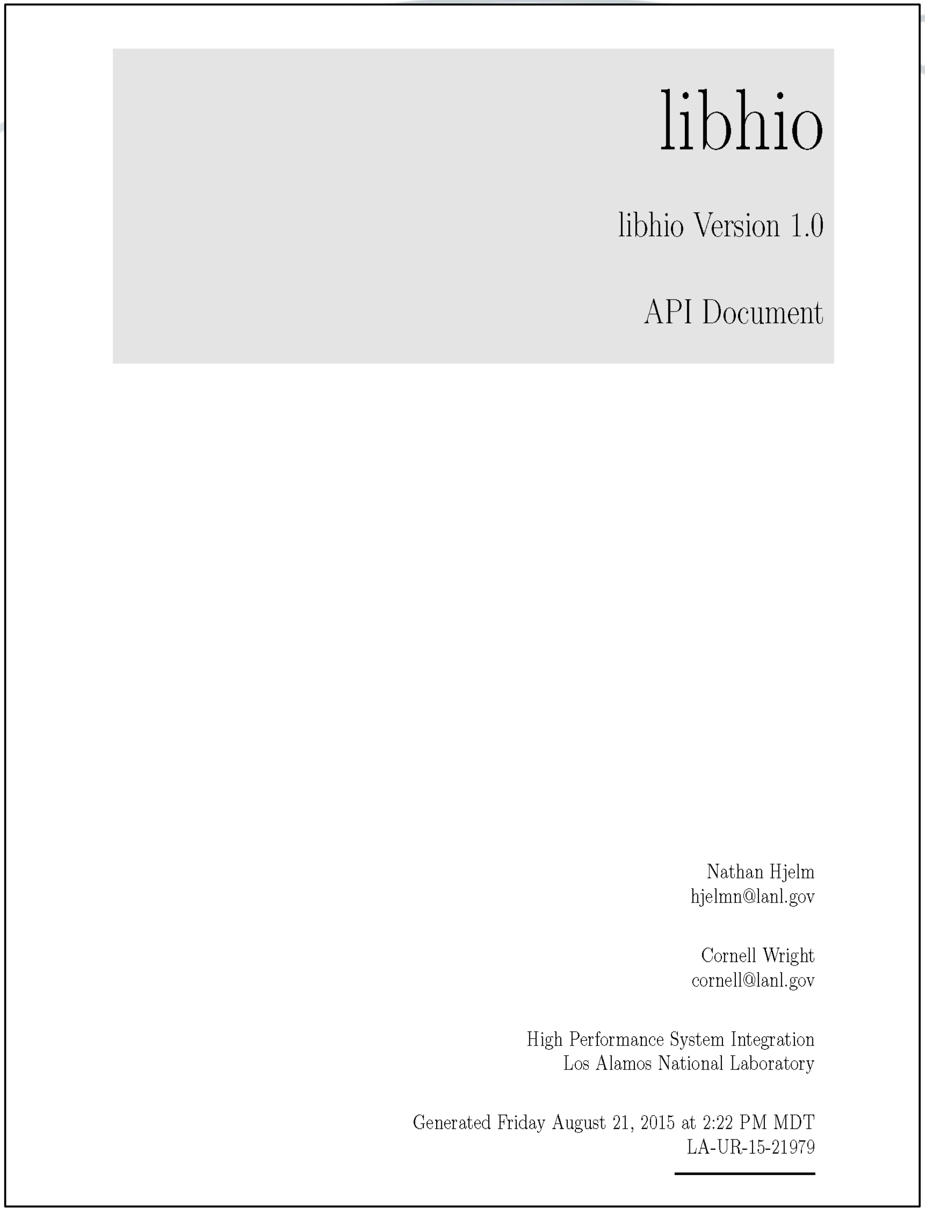
# hio – API Calls by Category

- **Context Management**
  - hio_init_single
  - hio_init_mpi
  - hio_fini
- **Checkpoint Management**
  - hio_dataset_should_checkpt
- **Data Collection Control**
  - hio_dataset_open
  - hio_dataset_close
  - hio_dataset_get_id
  - hio_dataset_unlink
  - hio_element_open
  - hio_element_close
  - hio_element_size
  - hio_dataset_construct
- **Data Movement**
  - hio_element_write
  - hio_element_write_nb
  - hio_element_write_strided
  - hio_element_write_strided_nb
  - hio_element_flush
  - hio_element_read
  - hio_element_read_nb
  - hio_element_read_strided
  - hio_element_read_strided_nb
  - hio_dataset_flush
- **Request Control**
  - hio_complete
  - hio_request_test
  - hio_request_wait
- **Configuration**
  - hio_config_set_value
  - hio_config_get_value
  - hio_config_get_count
  - hio_config_get_info
- **Performance Reporting**
  - hio_perf_get_count
  - hio_perf_get_info
  - hio_perf_get_value
- **Error Reporting**
  - hio_err_get_last
  - hio_err_print_last
  - hio_err_print_all

# HIO calls to DataWarp

| DataWarp APIs | Access DataWarp via POSIX |
|---|---|
| dw_stage_directory_out() | open(), fdopen() |
| dw_open_failed_stage() | fseek() |
| dw_read_failed_stage() | fread() |
| dw_wait_directory_stage() | fwrite() |
| dw_terminate_directory_stage() | access() |
| dw_query_directory_stage() | stat() |
| dw_set_stripe_configuration() | statfs()* |

* not actually POSIX



# libhio

libhio Version 1.0

API Document

Nathan Hjelm
hjelmn@lanl.gov

Cornell Wright
cornell@lanl.gov

High Performance System Integration
Los Alamos National Laboratory

Generated Friday August 21, 2015 at 2:22 PM MDT
LA-UR-15-21979

- libhio Document
  - API Document
  - User's Guide
- Library description
- HIO APIs
- Configuration variables
- 40 Pages

# Agenda

- Acknowledgements
- Burst Buffer Background
- Trinity Burst Buffer Hardware and Software
- HIO Design
- DataWarp Administration
- Multi-Job Scenarios
- Project Status

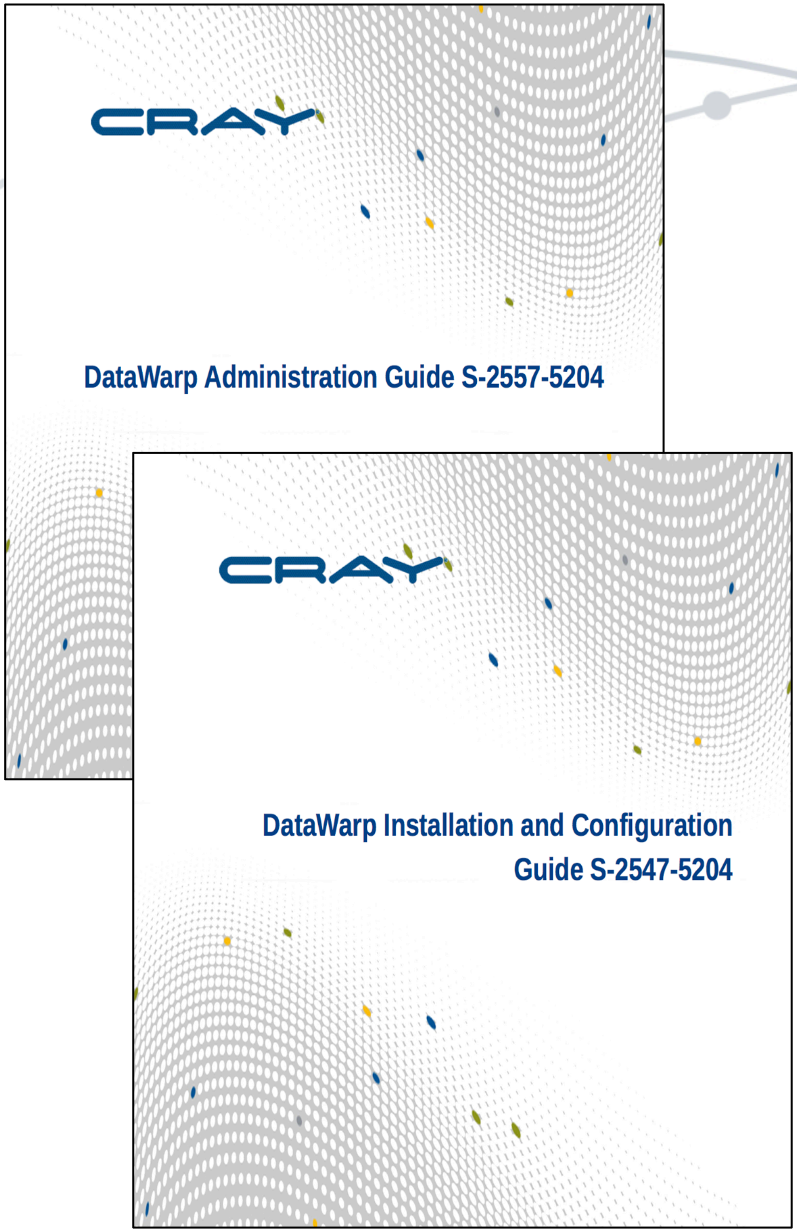
# Creating a Burst Buffer

- Normally performed by WLM
- Node SSD capacity in MiB: 7630912
  1/16th of that, rounded down to 16MiB multiple, in bytes: 500095254528  (about 465 GiB).
- Create pool with that granularity:
  ```
  -->module load dws
  -->dwcli create pool --name cw_pool --granularity 500095254528
  ```
- Add both DW nodes to pool:
  ```
  -->dwcli update node --name nid00017 --pool cw_pool
  -->dwcli update node --name nid00018 --pool cw_pool
  ```
- Create a session for all nodes:
  ```
  -->allhost="$(for i in $(apstat -n --no-summary --no-headers | sed  -e 's/^[ \t]*//' | cut -d ' ' -f 1); do printf "nid%.5d " $i; done)"
  -->dwcli create session --owner 4611 --creator cornell --token $RANDOM --hosts $allhost --expiration 0
  created session id 3
  ```
- Create instance with 16 allocation units (half of all space on the 2 nodes):
  ```
  -->dwcli create instance --session $sid --expiration 0 --label $RANDOM --capacity 8001524072448 --pool cw_pool
  created instance id 3
  ```
- Create a striped scratch configuration:
  ```
  -->dwcli create configuration --instance $iid --type scratch --access_type stripe --group 0 --root_permissions 0777
  created configuration id 3
  ```
- Create activation to mount at /tmp/dw_scr.  Note that parent directory must exist.
  ```
  -->dwcli create activation --mount /tmp/dw_scr --session $sid --configuration $cid
  created activation id 4
  ```

# DataWarp Status Command

- -->dwstat all

```
pool-id quantity     free granularity units
cw_pool 21.83TiB 14.55TiB   465.75GiB bytes
session-id token creator owner            created          expiration   goal nodes
         3  4156 cornell  4611 2015-08-21T13:10:32 1969-12-31T17:00:00 create    64
instance-id   bytes nodes             created          expiration expired intact label public session configurations
          3 7.28TiB     2 2015-08-21T13:14:32 1969-12-31T17:00:00   False   True  1127   True       3             1
configuration-id access_type activations instance    type
               3      stripe           1        3 scratch
registration-id configuration session wait
              3             3       3 True
activation-id configuration session nodes
            4             3       3    64
fragment-id capacity granularity instance     node missing
          4   953856        4MiB        3 nid00017   False
          5   953856        4MiB        3 nid00018   False
namespace-id configuration fragments
           3             3         2
 node-id    pool online granularity capacity instances activations
nid00017 cw_pool   True       16MiB  7.28TiB         1           0
nid00018 cw_pool   True       16MiB  7.28TiB         1           0
nid00020    None   True           0        0         0           1
nid00021    None   True           0        0         0           1
. . . .
nid00226    None   True           0        0         0           1
nid00227    None   True           0        0         0           1
nid00206 cw_pool   True       16MiB  7.28TiB         0           0
```



DataWarp Administration Guide S-2557-5204

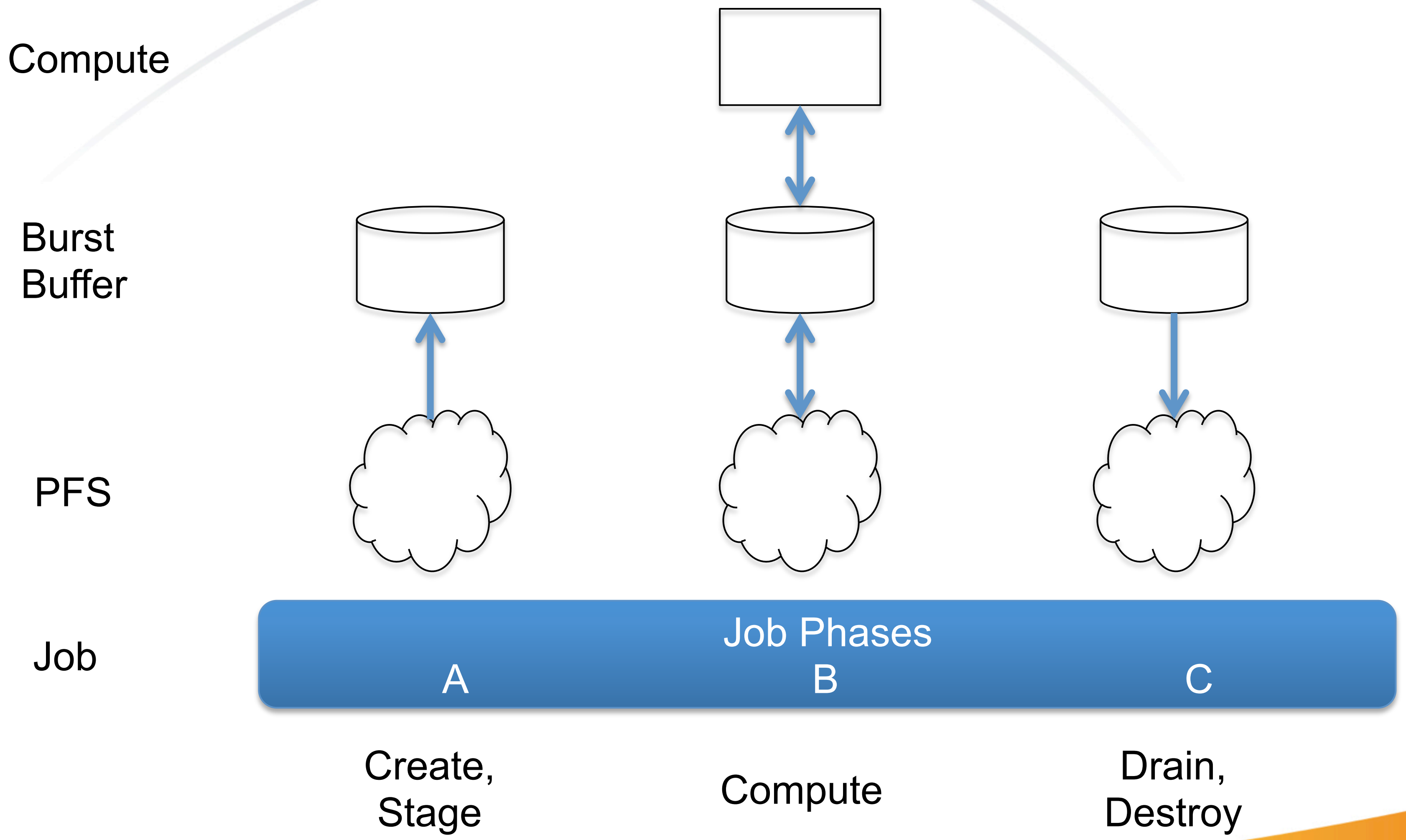
DataWarp Installation and Configuration Guide S-2547-5204

- DataWarp Administration Guide
  - Overview and concepts
  - Administrative commands
  - Administrator Tasks
  - Troubleshooting
  - Diagnostics
  - 50 Pages
- DataWarp Installation and Configuration
  - SSD Installation
  - Firmware maintenance
  - Configuration
  - Repair
  - 41 Pages

# Agenda

- Acknowledgements
- Burst Buffer Background
- Trinity Burst Buffer Hardware and Software
- HIO Design
- DataWarp Administration
- Multi-Job Scenarios
- Project Status

# Single-Job Burst Buffer



Compute

Burst Buffer

PFS

Job

Job Phases

A

B

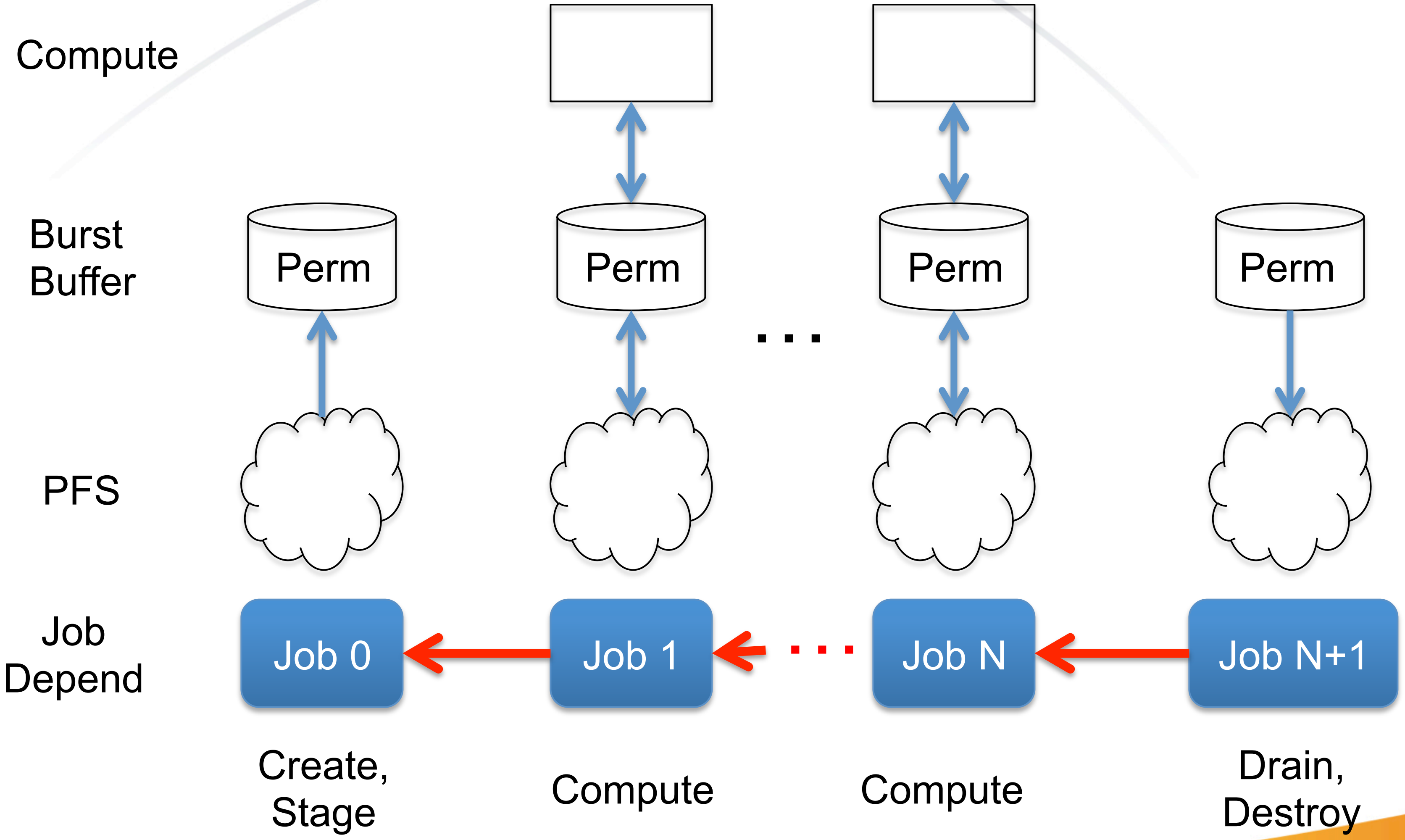
C

Create, Stage

Compute

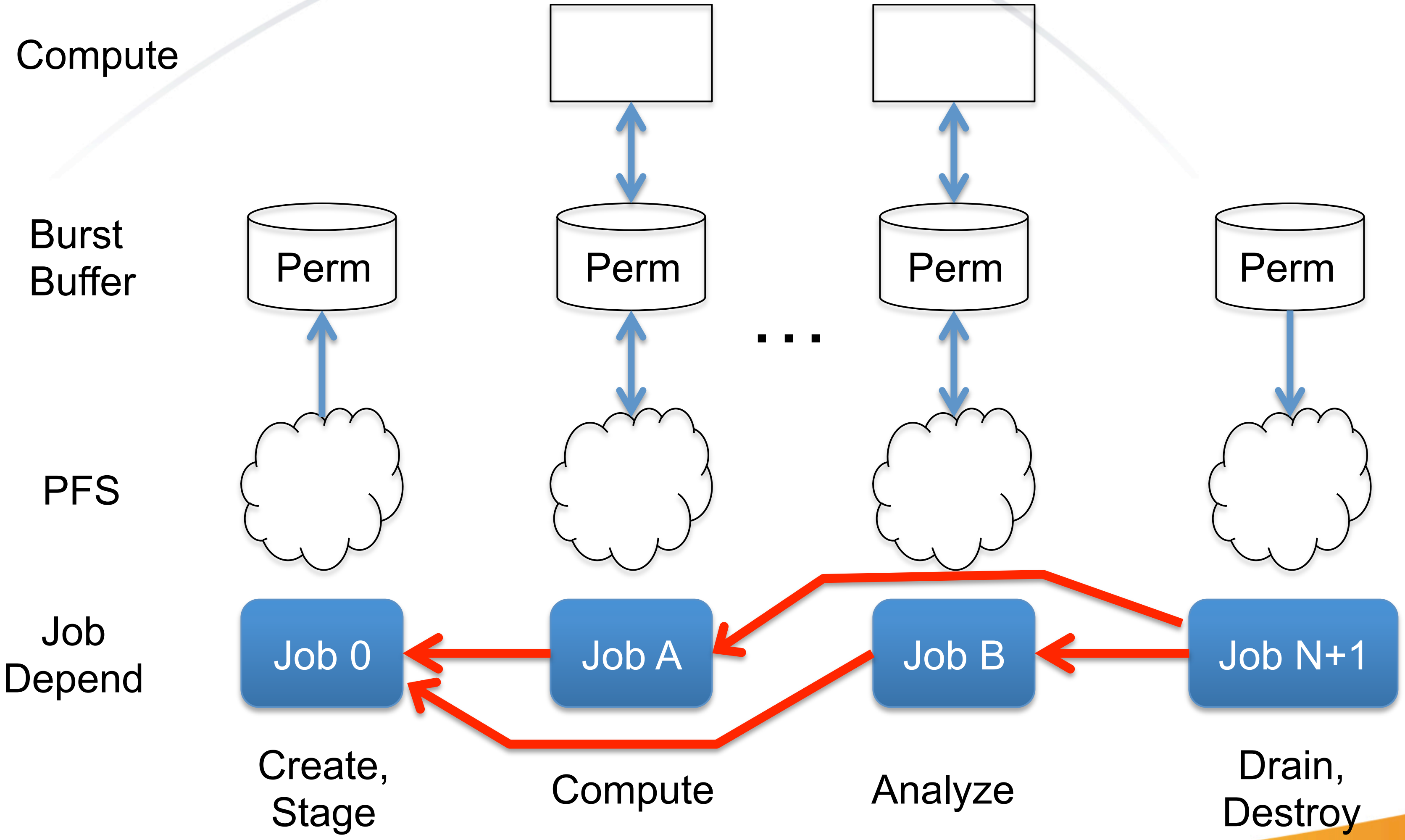
Drain, Destroy

# Multi-Job Burst Buffer



UNCLASSIFIED

# Compute + Analysis Burst Buffer



UNCLASSIFIED

# Agenda

- Acknowledgements
- Burst Buffer Background
- Trinity Burst Buffer Hardware and Software
- HIO Design
- DataWarp Administration
- Multi-Job Scenarios
- Project Status

# Development – Phase 1 DataWarp

- Phase 1 provides private and shared scratch (striped)
  - Meets checkpoint needs
  - Current release (UP04) currently lacking minor function, has performance deficiencies and bugs
- Installed on 2 LANL systems:
  - Gadget – 2 nodes with preproduction SSD
  - Buffy – 3 nodes with mixed preproduction / production SSD
- Trinitite – 6 nodes with production SSD
  - Will enable export controlled exposure to DataWarp
- WLM integration on Buffy now, Trinitite soon
- Medium scale testing on Cori (NERSC) in October
- Production SW Release:
  - Pearl/Pecos – 24 September 2015 (RC late August)
  - Rhine/Redwood – Early November 2015 ← Needed for Trinity
- Trinity – Product Level SSD cards (600 !) on site, install soon

# Development – Phase 2 DataWarp

- Phase 2 provides transparent and load balance caching
- Not required for checkpoint support
- Development version demonstrated July 2015
- Product release early 2016

# Development – HIO Library

- Major functions complete
  - Needs:
    - Checkpoint management
    - Optimized mode (code complete, in test)
- Tested with pre-production DataWarp and PFS
- API document LAUR'd and available
- Release 1.0 submitted to LANL Open Source process
- LANL code xRage ported to use HIO

# EAP Port to HIO: Phase 1

- Implemented via parallel module to bulkIO
  - Enabled via input deck option
  - Uses bulkIO IO distribution methodology
  - Uses HIO basic mode
- Full support for Cray DataWarp™ and PFS
- Some additional modification are needed outside of IO code
  - HIO “file” has a different name than expected
  - lastfile code can’t verify the file exists using inquire
  - Need to pass cycle number to module_hio for dataset identifier
- Checkpoint store and restart fully functional
  - Tested on Lustre PFS on Cielito, Cielo
  - Test with DataWarp on Trinitite late-September

# EAP Port to HIO: Phase 2

- Create parallel module to pio
- Requires HIO optimized mode for performance
- Does not use bulkio
  - Can be removed from code base in future
- Will provide equivalent or better performance and functionality as compared to module_pio
- Completion by end of CY'15

# Thank You !

# Questions